Error Term Checking: Towards Chosen Ciphertext Security without Re-encryption

Jan-Pieter D'Anvers and Emmanuela Orsini and Frederik Vercauteren

imec-COSIC KU Leuven
Kasteelpark Arenberg 10 - bus 2452, 3001 Leuven, Belgium
janpieter.danvers@esat.kuleuven.be,emmanuela.orsini@esat.kuleuven.be,
frederik.vercauteren@esat.kuleuven.be

Abstract. Chosen ciphertext security for lattice based encryption schemes is generally achieved through a generic transformation such as the Fujisaki-Okamoto transformation. This method requires full re-encryption of the plaintext during decapsulation, which typically dominates the cost of the latter procedure. In this work we show that it is possible to develop alternative transformations specifically designed for lattice based encryption schemes. We propose two novel chosen ciphertext transformations, `ETC1` and `ETC2`, in which re-encryption is replaced by checking the error term of the input ciphertext. We show that our new ciphertext validity check can be securely applied to lattice based encryption schemes under specific conditions. For the NIST post-quantum standardization candidate Threebears we show a speed-up for decapsulation of up to 37.4%. Moreover, as our method only changes the validation check during decapsulation, it is fully backwards compatible with existing implementations of the Fujisaki-Okamoto transformation.

1 Introduction

With the threat of quantum computers breaking the existing public key infrastructure, the development of replacement schemes that are secure against quantum computers is of paramount importance. These so called post-quantum cryptographic schemes have experienced an increased interest from both academia and industry in recent years. This interest has been further stimulated by the announcement of the NIST post-quantum standardization cryptography process [NIS16]. In the wake of this announcement several replacements for the current encryption and authentication schemes were proposed. At the moment of writing we are in the third round of this standardization process. One of the most practical and versatile families of quantum-safe primitives is based on hard problems on lattices, also known as lattice-based cryptography [Reg05,LPR10].

Public key encryption schemes directly derived from lattice problems typically only provide security against chosen-plaintext attacks. More advanced attackers that are capable of submitting chosen ciphertexts for decryption have been shown to be able to break these encryption schemes [JJ00,HNP+03,GN07,Flu16] For this reason, these schemes are often compiled to a chosen ciphertext secure key encapsulation mechanism (KEM) using a generic transformation based on the

Fujisaki-Okamoto (FO) transformation [FO13,Den03]. Classical and quantum secure versions of this transformation have been proposed with increasingly tighter security bounds in [TU16,HHK17,AOP+17,JZC+18,SXY18,BHH+19].

Decapsulation of a ciphertext in a FO compiled primitive roughly consists of two parts: decryption of the ciphertext into the message which was used as a random seed during encryption, and re-encryption of the plaintext into the valid ciphertext. The legitimacy of the input ciphertext is then checked by comparing the input ciphertext with the regenerated valid one.

While this is a generic off-the-shelf procedure that works for a wide range of encryption schemes, it can be rather expensive if the cost of re-encryption is higher than the cost of decryption. This is typically the case for schemes based on the learning with errors problem. The generic nature of this transformation also means that specific properties of the underlying encryption primitive are completely ignored. However, for other families of encryption schemes there exist specifically designed transformations exploiting properties of the underlying encryption that outperform the standard FO transformation, as is for example the case in RSA-OAEP+ [BR95,Sho01].

Our Contribution. Our goal is to show it is possible to design chosen ciphertext transformations tailored for lattice based cryptography by taking into account specific properties of the underlying encryption scheme. We propose a novel methodology to design chosen ciphertext secure KEMs out of lattice based encryption schemes and reduce the cost of validating an input ciphertext by replacing the generic check using a complete re-encryption of the ciphertext by a faster alternative.

Our approach is based on checking the validity of the error term, which is the consolidation of the noise introduced by both parties as a result of the learning with errors (LWE) paradigm. As such, the error term is a combination of the secret terms of the two communicating parties which makes it hard to predict. We argue that for some lattice based schemes it is hard, or sometimes even impossible, to construct a non-valid ciphertext with the **same error term** as the corresponding valid ciphertext of the same message.

Using this observation we develop two new chosen ciphertext transformations and bound their security. Our transformation relies on a new computational problem: Search Inner Product from Mod-LWE (SIP-LWE), in which an adversary is given l Mod-LWE samples $(\boldsymbol{A},\boldsymbol{b}:=\boldsymbol{A}\boldsymbol{s}+\boldsymbol{e})$ and is challenged to generate a new pair $(\boldsymbol{a}_*,\boldsymbol{b}_*)$ where $\boldsymbol{b}_*$ is the inner product of $\boldsymbol{a}_*$ and the secret key $\boldsymbol{s}$, i.e. $\boldsymbol{b}_*=\boldsymbol{a}_*^T\boldsymbol{s}$. Note that is equivalent to generating a Mod-LWE sample (or reusing a given one) with zero error. We also consider the special case where $\boldsymbol{b}_*$ is required to be $\boldsymbol{0}$, i.e. the problem now becomes generating a vector $\boldsymbol{a}_*$ that is orthogonal to $\boldsymbol{s}$. We denote this as the SIP0-LWE problem. We discuss the hardness of both problems, and show that for some parameter settings it is hard or even impossible for an adversary to distinguish the outcome of our transformation from a generic transformation with re-encryption.

Our transformation avoids the regeneration of the uniformly random public element of the public key Mod-LWE sample. Furthermore it changes the

number and the nature of the polynomial multiplications that are needed for the validity check. We show that we can securely apply these transformations to some encryption schemes accepted to the second round of the NIST Post Quantum Standardization Process, such as ThreeBears [Ham19], LAC [LLJ+19] and NewHope [PAA+19]. In particular for Threebears we implement our transformation and show it achieves a speedup of up to 37.4% over the generic FO transformation. Our transformation does not impact the communication between the parties or the encapsulation procedure, and can thus be used interchangeably with implementations based on the original NIST specifications. We conclude by noting that also the NTRU NIST post-quantum standardization process submission [ZCH+19], proposes a decapsulation algorithm that avoids re-encryption. However, their technique is fundamentally different from ours.

Paper Overview. After some preliminaries and security definitions in Section 2, we will give a general description of Mod-LWE encryption schemes in Section 2.3 and a discussion of the Fujisaki-Okamoto transformation in Section 2.4. In Section 3 we introduce the SIP-LWE and SIP0-LWE problem, which are used to construct two new transformations in Section 4. A comparison between the Fujisaki-Okamoto transformation and our newly introduced transformations is made in Section 5, including an implementation on the NIST submission ThreeBears.

2 Preliminaries

Let $\mathbb{Z}_q$ be the ring of integers modulo q and let R denote a ring, typically of the form $\mathbb{Z}_q$ or $\mathbb{Z}_q[X]/(X^n+1)$. $R^{l_1 \times l_2}$ will denote the ring of l_1 by l_2 matrices over R. Matrices will be written in uppercase bold letters, while vectors and ring elements will be written in lowercase bold letters. Let $\lfloor \cdot \rceil$ denote rounding to the closest integer and extend this notation coefficient-wise for matrices, vectors and polynomials.

Let $x \leftarrow \chi$ denote sampling x according to a distribution χ, which can be written more formally as $x \leftarrow \chi(R)$ to emphasize that $x \in R$. We can extend this operation coefficient-wise as $x \leftarrow \chi(R^{l_1 \times l_2})$, where each coefficient of $x \in R^{l_1 \times l_2}$ is sampled from the distribution χ. We denote by $x \leftarrow \chi(R;r)$ sampling x pseudorandomly according to the given seed r.

Let χ be a discrete distribution over a sample space S, the min-entropy of χ is a measure of how predictable the distribution is.

Definition 1. *Let χ be a discrete distribution. The* min-entropy *of χ is defined as:*

$$H_\infty(\chi) = -\log(\max_x \Pr[x | x \leftarrow \chi]).$$

We recall a basic property of min-entropy.

Proposition 1. *Let χ_1 and χ_2 be two independent distributions, then:*

$$H_\infty(\chi_1 + \chi_2) \geq \max(H_\infty(\chi_1), H_\infty(\chi_2))$$

2.1 Cryptographic Definitions

A *public-key encryption scheme* PKE consists of three PPT algorithms (KeyGen, Enc, Dec). The key generation algorithm KeyGen takes as input a security parameter 1^n and generates a public/secret key-pair (pk,sk); the encryption algorithm Enc outputs a ciphertext c when given the public key pk and a message $m \in \mathcal{M}$, where $\mathcal{M}$ is the message space; and the decryption algorithm Dec uses the secret key sk and a ciphertext c to retrieve a message m^* or a failure message $\perp$. Looking ahead, we will sometimes make explicit the randomness used for encrypting by writing $c := \mathsf{Enc}(pk,m;r)$, where $r \in \mathcal{R}$ and $\mathcal{R}$ denotes the randomness space. A PKE scheme is said to be δ-*correct* if:

$$E\left[\max_{m \in \mathcal{M}} \Pr[\mathsf{Dec}(sk,c) \neq m : c \leftarrow \mathsf{Enc}(pk,m)]\right] \leq \delta,$$

where the expectation is taken over $(pk,sk) \leftarrow \mathsf{KeyGen}$ and the randomness of the encryption procedure $\mathsf{Enc}(pk,m)$. The scheme is perfectly correct if $\delta = 0$.

We recall some basic security definitions for PKE schemes.

Definition 2 (OW-CPA). *Let* PKE *be a public-key encryption scheme with message space* $\mathcal{M}$*. We define the* OW-CPA *advantage of an adversary* $\mathcal{A}$ *as follows:*

$$\mathtt{Adv}^{ow\text{-}cpa}_{\mathsf{PKE}}(\mathcal{A}) = \Pr\left[m = m^* : \begin{array}{c}(pk,sk) \leftarrow \mathsf{KeyGen}();\ m \leftarrow \mathcal{M};\\ c \leftarrow \mathsf{Enc}(pk,m);\ m^* \leftarrow \mathcal{A}^{\mathsf{Enc}}(pk,c);\end{array}\right].$$

Definition 3 (IND-CPA). *Let* PKE *be a public-key encryption scheme with message space* $\mathcal{M}$*. We define the* IND-CPA *advantage of an adversary* $\mathcal{A}$ *as follows:*

$$\mathtt{Adv}^{ind\text{-}cpa}_{\mathsf{PKE}}(\mathcal{A}) = \left|\Pr\left[b = b^* : \begin{array}{c}(pk,sk) \leftarrow \mathsf{KeyGen}();\\ m_0,m_1 \leftarrow \mathcal{A}(pk), m_0,m_1 \in \mathcal{M};\\ b \leftarrow \{0,1\}; c_b \leftarrow \mathsf{Enc}(pk,m_b);\\ b^* \leftarrow \mathcal{A}^{\mathsf{Enc}}(pk,c_b)\end{array}\right] - \frac{1}{2}\right|.$$

A *key encapsulation mechanism* (KEM) consists of three algorithms (KeyGen, Encaps, Decaps). The key generation algorithm KeyGen generates a key pair (pk,sk), the encapsulation algorithm Encaps generates a ciphertext c and a key k on input the public key pk, and the decapsulation algorithm Decaps, on input the secret key sk and an encapsulation c, outputs a key k'.

Similarly to above, the KEM is said to be δ-correct if:

$$\Pr[\mathsf{Decaps}(sk,c) \neq k : (c,k) \leftarrow \mathsf{Encaps}(pk)] \leq \delta.$$

The IND-CCA security notion for a KEM is given below, where $\mathcal{A}^{\mathsf{Decaps}}$ signifies that the adversary $\mathcal{A}$ is given access to a decapsulation oracle, with the restriction that the input ciphertext is not the challenge ciphertext c.

Definition 4 (IND-CCA). *Let* KEM *be a key encapsulation mechanism with key space $\mathcal{K}$. We define the* IND-CCA *advantage of an adversary $\mathcal{A}$ as:*

$$\mathtt{Adv}^{ind\text{-}cca}_{\mathsf{KEM}}(\mathcal{A}) = \left| \Pr\left[b' = b : \begin{array}{c} (pk,sk) \leftarrow \mathsf{KeyGen}(); \\ b \leftarrow \mathcal{U}(\{0,1\}); \\ (c,k_0) \leftarrow \mathsf{Encaps}(pk); k_1 \leftarrow \mathcal{K}; \\ b' \leftarrow \mathcal{A}^{\mathsf{Decaps}}(pk,c,k_b); \end{array} \right] - \frac{1}{2} \right|.$$

For our security proof, we will also need a measure of the entropy of a function. To this end we now extend the definition of γ-uniformity introduced by Fujisaki and Okamoto [FO13] to any pseudorandom function f that takes as input a public key pk, a secret key sk and a message m, and uses r as a seed for its randomness. The γ-uniformity gives a measure of how predictable the output of f is for a random seed r. The *min-entropy* of an encryption function $f(pk,sk,m;r)$ is defined as:

$$\gamma(pk,sk,m) := -\log\left(\max_{y \in Y} \Pr_{r \in R}[y = f(pk,sk,m;r)] \right).$$

We say that f is γ-spread if for every $(pk,sk) \leftarrow \mathsf{KeyGen}()$ and for every $m \in \mathcal{M}$, $\gamma(pk,sk,m) \geq \gamma$.

2.2 Module Learning with Errors

The Module Learning with Errors (Mod-LWE) problem generalizes both the Learning with Errors (LWE) problem introduced by Regev [Reg05] and Ring Learning with errors (Ring-LWE) [SSTX09,LPR10] problem. Here we recall both its decision and search variant.

Definition 5. *Let l be a positive integer, R a ring, and χ_s and χ_e two probability distributions on R with limited variance. Generate the secret elements $\boldsymbol{s}$ and $\boldsymbol{e}$ following the distributions $\chi_s(R^{l\times 1})$ and $\chi_e(R^{l\times 1})$ respectively, and generate the public element $\boldsymbol{A}$ uniformly from $\mathcal{U}(R^{l\times l})$. Denote with $\mathcal{L}$ the distribution of Mod-LWE samples $(\boldsymbol{A}, \boldsymbol{As}+\boldsymbol{e})$ generated using the aforementioned procedure.*

- *The* Decision Mod-LWE problem *is to distinguish a Mod-LWE sample $(\boldsymbol{A},\boldsymbol{b}) \leftarrow \mathcal{L}$ from a uniformly random sample $(\boldsymbol{A},\boldsymbol{u}) \leftarrow \mathcal{U}(R^{l\times l} \times R^{l\times 1})$. More formally, we define the advantage of an adversary $\mathcal{A}$ against this problem as:*

$$\mathtt{Adv}^{\mathsf{decision}}_{Mod\text{-}LWE}(A) = P\left[c = c^* : \begin{array}{c} \boldsymbol{A} \leftarrow \mathcal{U}(R^{l\times l});\ \boldsymbol{s} \leftarrow \chi_s(R^{l\times 1}); \\ \boldsymbol{e} \leftarrow \chi_e(R^{l\times 1});\ \boldsymbol{b}_0 = \boldsymbol{As}+\boldsymbol{e}; \\ \boldsymbol{b}_1 \leftarrow \mathcal{U}(R^{l\times 1});\ c \leftarrow \mathcal{U}(\{0,1\}); \\ c^* \leftarrow A(\boldsymbol{A},\boldsymbol{b}_i); \end{array} \right].$$

- *The* Search Mod-LWE problem *consists of finding the secret $\boldsymbol{s}$ from a Mod-LWE sample $(\boldsymbol{A},\boldsymbol{b}) \leftarrow \mathcal{L}$. The advantage of an adversary $\mathcal{A}$ against this problem is defined as:*

$$\mathtt{Adv}^{\mathsf{search}}_{Mod\text{-}LWE}(A) = P\left[\boldsymbol{s} = \boldsymbol{s}^* : \begin{array}{c} \boldsymbol{A} \leftarrow \mathcal{U}(R^{l\times l}); \boldsymbol{s} \leftarrow \chi_s(R^{l\times 1}); \\ \boldsymbol{e} \leftarrow \chi_e(R^{l\times 1}); \boldsymbol{b} = \boldsymbol{As}+\boldsymbol{e}; \\ \boldsymbol{s}^* \leftarrow A(\boldsymbol{A},\boldsymbol{b}); \end{array} \right].$$

Note that the Ring-LWE hard problem is a specific case of the Mod-LWE problem where $l=1$.

2.3 Mod-LWE Based Encryption.

We now describe a PKE scheme based on Mod-LWE. Let R be a ring and gen a deterministic function that expands a seed $seed_{\boldsymbol{A}} \in \{0,1\}^{l_m}$ to a uniformly distributed matrix $\boldsymbol{A} \in R^{l \times l}$. Let χ_s and χ_e be two distributions over R with limited variance, and $\mathcal{M}=\{0,1\}^{l_m}$ and $\mathcal{R}=\{0,1\}^{l_r}$ be the message and randomness space respectively.

Let encode be a function that encodes $m \in \mathcal{M}$ into a ring element $\boldsymbol{y}$, and decode be a function that takes a ring element $\boldsymbol{y}'=\boldsymbol{y}+\boldsymbol{e}_{\text{tot}}$, where $\boldsymbol{e}_{\text{tot}}$ is a small ring element in some sense, and that outputs $m' \in \mathcal{M}$ which equals m with high probability. Typically, when working in the polynomial ring $R=\mathbb{Z}_q[X]/(X^N+1)$, encode interpretes m as a ring element where the i^{th} coefficient of the polynomial is the i^{th} bit of m. The output of the encode function is then computed as $\boldsymbol{y}=\lfloor q/2 \cdot m \rfloor$; on the contrary, the decode function computes $m'=\lfloor 2/q \cdot \boldsymbol{y}' \rceil$ and returns the bitstring interpretation of m'.

We also define two more functions $\mathsf{compress}$ and $\mathsf{decompress}$ that "compress" and "decompress" a ring element $\boldsymbol{v}'_m \in R$ respectively. This procedure introduces an error $\boldsymbol{u}$, which can be calculated as follows:

$$\boldsymbol{u}=\boldsymbol{v}'_m-\mathsf{decompress}(\mathsf{compress}(\boldsymbol{v}'_m))$$

.

More precisely, given a ring $R=\mathbb{Z}_q[X]/(X^N+1)$, compression is typically done by rounding off lower significant bits. In this case, for $p<q$, compression and decompression can be defined as follows:

$$\begin{aligned}\mathsf{compress}(\boldsymbol{v}'_m)&=\lfloor p/q \cdot \boldsymbol{v}'_m \rceil\\ \mathsf{decompress}(\boldsymbol{v}'_m)&=\lfloor q/p \cdot \boldsymbol{v}'_m \rceil\end{aligned}$$

Using the aforementioned functions and parameters, a PKE can be constructed as defined in Figure 1.

Looking ahead, we notice that when truthfully executing the encryption scheme, the difference between $\boldsymbol{v}'_d$ and $\boldsymbol{v}$ can be calculated as:

$$\begin{aligned}\boldsymbol{v}'_d-\boldsymbol{v}&=\mathsf{decompress}(\mathsf{compress}(\boldsymbol{v}'_m))-\boldsymbol{b}'^T\boldsymbol{s}\\ &=\boldsymbol{v}'_m+\boldsymbol{u}-\boldsymbol{s}'^T\boldsymbol{A}\boldsymbol{s}-\boldsymbol{e}'^T\boldsymbol{s}\\ &=\boldsymbol{s}'^T\boldsymbol{A}\boldsymbol{s}+\boldsymbol{s}'^T\boldsymbol{e}+\boldsymbol{e}''+\mathtt{encode}(m)+\boldsymbol{u}-\boldsymbol{s}'^T\boldsymbol{A}\boldsymbol{s}-\boldsymbol{e}'^T\boldsymbol{s}\\ &=\boldsymbol{s}'^T\boldsymbol{e}-\boldsymbol{e}'^T\boldsymbol{s}+\boldsymbol{e}''+\boldsymbol{u}+\mathtt{encode}(m).\end{aligned}$$

Therefore, the decryption is correct if the error term $\boldsymbol{e}_{\text{tot}}=\boldsymbol{s}'^T\boldsymbol{e}-\boldsymbol{e}'^T\boldsymbol{s}+\boldsymbol{e}''+\boldsymbol{u}$ is small in some scheme-dependent sense. If the error term $\boldsymbol{e}_{\text{tot}}$ is too large a decryption failure will occur. If the PKE scheme is δ-correct, this happens with probability at most δ, which is typically lower then 2^{-128}.

$\mathsf{PKE.KeyGen}()$

1. $\text{seed}_{\boldsymbol{A}} \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
2. $\boldsymbol{A} := \mathsf{gen}(\text{seed}_{\boldsymbol{A}}) \in R^{l \times l}$
3. $\boldsymbol{s} \leftarrow \chi_s(R^{l \times 1})$
4. $\boldsymbol{e} \leftarrow \chi_e(R^{l \times 1})$
5. $\boldsymbol{b} := \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}$
6. $pk := (\boldsymbol{b}, seed_{\boldsymbol{A}})$
7. $sk := (\boldsymbol{s}, \boldsymbol{e})$
8. **return** (pk, sk)

$\mathsf{PKE.Dec}(sk, c)$

1. $\boldsymbol{v} := \boldsymbol{b'}^T \boldsymbol{s}$
2. $\boldsymbol{v}'_d := \mathsf{decompress}(\boldsymbol{v}'_c)$
3. $\overline{m} := \mathsf{decode}(\boldsymbol{v}'_d - \boldsymbol{v})$
4. **return** $\overline{m}$

$\mathsf{PKE.Enc}(pk = (\boldsymbol{b}, \text{seed}_{\boldsymbol{A}}), m; r)$

1. $\boldsymbol{A} := \mathsf{gen}(\text{seed}_{\boldsymbol{A}}) \in R^{l \times l}$
2. $\boldsymbol{s}' \leftarrow \chi_s(R^{l \times 1}; r)$
3. $\boldsymbol{e}' \leftarrow \chi_e(R^{l \times 1}; r)$
4. $\boldsymbol{e}'' \leftarrow \chi_e(R^{1 \times 1}; r)$
5. $\boldsymbol{b}' := \boldsymbol{A}^T \boldsymbol{s}' + \boldsymbol{e}'$
6. $\boldsymbol{v}' := \boldsymbol{b}^T \boldsymbol{s}'$
7. $\boldsymbol{v}'_m := \boldsymbol{v}' + \boldsymbol{e}'' + \mathsf{encode}(m)$
8. $\boldsymbol{v}'_c := \mathsf{compress}(\boldsymbol{v}'_m)$
9. **return** $c := (\boldsymbol{v}'_c, \boldsymbol{b}')$

Fig. 1: Mod-LWE-based PKE

2.4 Fujisaki-Okamoto Transformation

As detailed in the introduction, the encryption scheme described in the previous section (Section 2.3) is not secure against chosen ciphertext attacks. To achieve IND-CCA security, typically the PKE scheme is transformed into a KEM using a post-quantum variant [TU16,HHK17,JZC+18,SXY18,BHH+19]of the Fujisaki-Okamoto (FO) [FO13] transformation. This is a general technique which is valid for any encryption scheme, and thus not necessarily optimized for lattice based schemes.

The key generation function $\mathsf{KEM}_{\mathsf{FO}}.\mathsf{KeyGen}$ generates the public key and secret key as in $\mathsf{PKE.KeyGen}$, but includes an additional uniformly random value $w \in \mathcal{M}$ in the secret key, which is used during the decapsulation of a non-valid ciphertext. Let $G : \mathcal{M} \times R^{l \times 1} \rightarrow \mathcal{R}$ and $H : \mathcal{M} \times \mathcal{C} \rightarrow \mathcal{K}$ be two hash functions modeled as random oracles. Then the functions Encaps and Decaps can be constructed as given in Figure 2, respectively.

$\mathsf{KEM}_{\mathtt{FO}}.\mathsf{Encaps}(pk)$

1. $m \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
2. $r := G(m, pk)$
3. $c := \mathsf{PKE.Enc}(pk, m; r)$
4. $K := H(m, c)$
5. **return** (c, K)

$\mathsf{KEM}_{\mathtt{FO}}.\mathsf{Decaps}(sk = (\boldsymbol{s}, \boldsymbol{e}, w), pk = (\boldsymbol{b}, \text{seed}_{\boldsymbol{A}}), c = (\boldsymbol{v}'_c, \boldsymbol{b}'))$

1. $\overline{m} := \mathsf{PKE.Dec}(sk, c)$ `// decrypt`
2. $\overline{r} := G(\overline{m}, pk)$
3. $\overline{c} := \mathsf{PKE.Enc}(pk, \overline{m}; \overline{r})$ `// re-encrypt`
4. **if** $c = \overline{c}$ `// check ciphertexts`
5. **return** $K := H(\overline{m}, c)$
6. **else**
7. **return** $K := H(w, c)$

Fig. 2: IND-CCA secure KEM $\mathtt{FO}^{\not\perp}$

In essence, the FO transformation during the decapsulation phase checks whether a valid ciphertext c was submitted, by re-encrypting the decrypted message $\overline{m}$ to the ciphertext $\bar{c}$, and by subsequently checking whether the result matches the submitted ciphertext c. For the rest of the paper we will denote with $\bar{\cdot}$ variables that are regenerated based on the decrypted message $\overline{m}$. The cost of this re-encryption is typically dominated by the generation of the matrix $\boldsymbol{A}$ and $(l+1)l$ multiplications of elements in R.

There are two main variants of this transformation: explicit rejection, denoted $\texttt{FO}^{\perp}$, where $\perp$ is returned when the validation check fails, and implicit rejection, denoted with $\texttt{FO}^{\not\perp}$, where a pseudorandom value is returned on a failing check. In this paper we will focus on the latter as it is the version generally used in the relevant NIST schemes. In the rest of the paper we will sometimes use FO to refer to $\texttt{FO}^{\not\perp}$. It should be noted that our methodology can also be applied to $\texttt{FO}^{\perp}$.

The IND-CCA security of this transformation can be bound using Theorem 3.1 and Theorem 3.3 of Hofheinz et al. [HHK17] as follows:

Theorem 1 (PKE OW-CPA $\overset{\text{ROM}}{\Rightarrow}$ KEM IND-CCA)**.** *Given a δ-correct PKE, for any classical adversary $\mathcal{A}$ against the IND-CCA security of $\mathsf{KEM}_{\texttt{FO}}$ issuing at most q_D queries to the decapsulation oracle and at most q_G and q_H queries to the random oracles $\mathcal{G}$ and $\mathcal{H}$ respectively, there exists a OW-CPA adversary $\mathcal{B}$ against PKE such that:*

$$\texttt{Adv}^{ind\text{-}cca}_{\texttt{FO}}(\mathcal{A}) \leq q_H/2^{l_m} + q_G\delta + (q_G+1)\texttt{Adv}^{ow\text{-}cpa}_{\mathsf{PKE}}(\mathcal{B}),$$

where the running time of $\mathcal{A}$ is not much higher than the running time of $\mathcal{B}$.

3 SIP-LWE

The core idea underlying our new IND-CCA secure transformations is as follows: when truthfully executing the algorithm, $\boldsymbol{v}'_d - \boldsymbol{v}$ equals the term $\boldsymbol{s}'^T\boldsymbol{e} - \boldsymbol{e}'^T\boldsymbol{s} + \boldsymbol{e}'' + \boldsymbol{u} + \texttt{encode}(m)$, which we will denote by $\boldsymbol{z}$. Instead of performing a full re-encryption and verification of the ciphertext, our new transformation will simply check whether $\boldsymbol{v}'_d - \boldsymbol{b}'^T\boldsymbol{s} = \boldsymbol{z}$.

To bound the security of our two new transformations, we rely on a new computational problem that we call *Search Inner Product from Mod-LWE problem* (SIP-LWE). In the SIP-LWE problem, the adversary is asked to generate a vector $\boldsymbol{a}_*$ and a ring element $\boldsymbol{b}_*$, where $\boldsymbol{b}_*$ should be the inner product of $\boldsymbol{a}_*$ and the secret $\boldsymbol{s}$ of the Mod-LWE samples, i.e. $\boldsymbol{a}_*^T\boldsymbol{s} = \boldsymbol{b}_*$. A special case of the SIP-LWE problem is where $\boldsymbol{b}_* = \boldsymbol{0}$, i.e. the adversary has to generate a vector $\boldsymbol{a}_*$ that is orthogonal to $\boldsymbol{s}$; we denote this restricted problem as SIP0-LWE. Note that the SIP-LWE problem can also be viewed as generating a Mod-LWE sample with zero error term $(\boldsymbol{a}_*, \boldsymbol{b}_* = \boldsymbol{a}_*^T\boldsymbol{s})$.

We will show that the security of the newly introduced ETC1 and ETC2 transformations can be bound by the hardness of the two variants of the SIP-LWE problem. Intuitively, an adversary that tries to break the security of the ETC1

transformation wants to adapt $\boldsymbol{v}'_d$ and $\boldsymbol{b}'$ so that $\boldsymbol{v}'_d - \boldsymbol{b}'^T\boldsymbol{s} = \boldsymbol{z}$. We will show that this boils down to finding an $\boldsymbol{a}_*$, $\boldsymbol{b}_*$, perturbation to $\boldsymbol{b}'$ and $\boldsymbol{v}'_d$, so that:

$$(\boldsymbol{v}'_d + \boldsymbol{b}_*) - (\boldsymbol{b}' + \boldsymbol{a}_*)'^T\boldsymbol{s} = \boldsymbol{s}'^T\boldsymbol{e} - (\boldsymbol{e}' + \boldsymbol{a}_*)^T\boldsymbol{s} + \boldsymbol{e}'' + \boldsymbol{u} + \boldsymbol{b}_*$$

equals the original error term:

$$\boldsymbol{z} = \boldsymbol{s}'^T\boldsymbol{e} - \boldsymbol{e}'^T\boldsymbol{s} + \boldsymbol{e}'' + \boldsymbol{u}.$$

One can see from equating these error terms that this challenge can be rephrased as generating a pair $(\boldsymbol{a}_*, \boldsymbol{b}_*)$ so that $\boldsymbol{b}_* = \boldsymbol{a}_*^T\boldsymbol{s}$. In case of the `ETC2` transformation, there are additional checks in place that force $\boldsymbol{b}_* = 0$ so that an adversary has to find $\boldsymbol{a}_*^T\boldsymbol{s} = 0$.

3.1 The SIP-LWE problem

The central tool of our transformations is the SIP-LWE problem, which is formally defined as follows:

Definition 6 (SIP-LWE problem). *Let l be a positive integer, R be an arbitrary ring and χ_s, χ_e be two distributions on R. Let $\boldsymbol{s} \leftarrow \chi_s(R^{l\times 1})$, $\boldsymbol{e} \leftarrow \chi_e(R^{l\times 1})$ and $\boldsymbol{A} \leftarrow \mathcal{U}(R^{l\times l})$ sampled uniformly at random. The* Search Inner Product *from Mod-LWE problem (SIP-LWE$_{R,l,\chi_s,\chi_e}$) problem consists of finding a non-zero vector $\boldsymbol{a}_* \in R^l$ and a ring element $\boldsymbol{b}_* \in R$ such that $\boldsymbol{a}_*^T\boldsymbol{s} = \boldsymbol{b}_*$, where $\boldsymbol{s}$ is the secret of the l given Mod-LWE samples $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}+\boldsymbol{e})$. The advantage of an adversary $\mathcal{A}$ is defined below:*

$$\mathtt{Adv}^{\textit{SIP-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A}) = P\left[\begin{array}{c} \boldsymbol{a}_*^T\boldsymbol{s} = \boldsymbol{b}_* \quad \boldsymbol{s} \leftarrow \chi_s(R^{l\times 1});\ \boldsymbol{e} \leftarrow \chi_e(R^{l\times 1}) \\ \textit{and} \quad : \quad \boldsymbol{A} \leftarrow \mathcal{U}(R^{l\times l});\ \boldsymbol{b} := \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e}; \\ \boldsymbol{a}_* \neq 0 \qquad\quad (\boldsymbol{a}_*, \boldsymbol{b}_*) \leftarrow \mathcal{A}(\boldsymbol{A}, \boldsymbol{b}) \end{array}\right].$$

Definition 7 (SIP0-LWE problem). *The SIP0-LWE problem is the SIP-LWE problem with the extra restriction that a solution is valid only when $\boldsymbol{b}_* = \boldsymbol{0}$.*

Notice, since any solution to the SIP0-LWE problem is a valid solution to the SIP-LWE problem by setting $(\boldsymbol{b}_* = 0)$, it is clear that

$$\mathtt{Adv}^{\text{SIP0-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A}) \leq \mathtt{Adv}^{\text{SIP-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A}),$$

so to get an upper bound it suffices to analyze the SIP-LWE problem. However, in specific cases, the SIP0-LWE problem is much harder than the SIP-LWE problem, to the extent that it is impossible to solve, as we now illustrate.

Hardness of rank one SIP(0)-LWE. First, consider the hardness of the rank one SIP0-LWE problem, which challenges an adversary to find a non-zero ring element $\boldsymbol{a}_* \in R$ such that $\boldsymbol{a}_*\boldsymbol{s} = 0$. This clearly is only solvable when $\boldsymbol{s}$ is non-invertible, since if $\boldsymbol{s}$ is invertible, the equation would imply $\boldsymbol{a}_* = 0$ which is not a valid solution. We therefore conclude that for invertible $\boldsymbol{s}$, rank one SIP0-LWE

simply has no solution, and thus the advantage of any adversary is 0. Note that this holds for all rings, in particular also for NTT rings (i.e. where $f(x)$ splits completely modulo q) when one restricts to invertible secrets $\boldsymbol{s}$.

Similarly, it is easy to see that solving rank one SIP-LWE when R is a field, is equivalent to solving the Search-Mod-LWE problem: indeed, any non-zero $\boldsymbol{a}_*$ and element $\boldsymbol{b}_*$ such that $\boldsymbol{a}_*\boldsymbol{s}=\boldsymbol{b}_*$, immediately gives $\boldsymbol{s}=\boldsymbol{b}_*\boldsymbol{a}_*^{-1}$.

3.2 Solving the SIP-LWE problem

The SIP-LWE problem asks to find a vector $\boldsymbol{a}_*$ with known inner product $\boldsymbol{b}_*$, i.e. $\boldsymbol{a}_*^T\boldsymbol{s}=\boldsymbol{b}_*$, given additional information about $\boldsymbol{s}$ via the l Mod-LWE samples. We first describe three simple approaches that already result in lower bounds on $\mathtt{Adv}^{\text{SIP-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A})$, and show how they can be generalised depending on the structure of the ring R.

The first approach is to simply solve the Search Mod-LWE problem first to recover $\boldsymbol{s}$, and to return a random vector $\boldsymbol{a}_*$ together with $\boldsymbol{b}_*=\boldsymbol{a}_*^T\boldsymbol{s}$, which shows that

$$\mathtt{Adv}^{\text{SIP-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A}) \geq \mathtt{Adv}^{\text{Search-Mod-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{B}),$$

for all adversaries $\mathcal{B}$.

A second approach to solving the SIP-LWE problem is to use the given Mod-LWE samples to derive a solution: since we have a description of χ_e we can select the most likely ring element $\boldsymbol{c}$ that would have be sampled from $\chi_e(R)$ (typically this will be the zero element), and return as solution $(\boldsymbol{a},\boldsymbol{b}-\boldsymbol{c})$, where $(\boldsymbol{a},\boldsymbol{b})$ is one of the given Mod-LWE samples. This immediately shows that

$$\mathtt{Adv}^{\text{SIP-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A}) \geq 2^{-H_\infty(\chi_e(R))}.$$

In particular, this shows that the SIP-LWE problem is *easy* for typical instances where the ring R is rather small (and thus the rank l is large), e.g. like in standard LWE.

The third approach is to make a similar reasoning on χ_s, by ignoring the given Mod-LWE samples, and choosing a vector $\boldsymbol{\delta}_i$ which has all zeros expect for a one in position $0<i\leq l$, and guessing the i-th secret key element $\boldsymbol{s}_i$ as the most likely element $\boldsymbol{d}$ to be sampled from $\chi_s(R)$. The proposed solution then is $(\boldsymbol{\delta}_i,\boldsymbol{d})$ which gives the lower bound

$$\mathtt{Adv}^{\text{SIP-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A}) \geq 2^{-H_\infty(\chi_s(R))}.$$

Note that these two approaches show that for small rings R (e.g. typically encountered when the rank l is large) , the SIP-LWE problem is in fact easy.

Exploiting the structure of R. Depending on the structure of the ring R the above approaches can be generalised to factors of R. For simplicity we assume that R is of the form $\mathbb{Z}_q[X]/(f(X))$ with q a prime and $f(X)$ a monic polynomial of degree d which is square-free modulo q. Let the factorisation of

$f(X)$ be $\prod_{i=1}^{t} f_i(X)$ with f_i irreducbile over $\mathbb{Z}_q[X]$, then the Chinese remainder theorem says that $R \cong R_1 \times \cdots R_t$ with $R_i = \mathbb{Z}_q[X]/(f_i(X))$, and since the f_i are irreducible the R_i are now fields. Furthermore, for each i we also have the projection map π_i, which maps R into R_i by killing off all other components R_j. In particular, this projection map is simply multiplication by the Lagrange multiplier $\lambda_i = u_i(x) \cdot (u_i(x)^{-1} \bmod f_i(x))$ where $u_i(x) = f(x)/f_i(x)$.

If an adversary can solve SIP-LWE for the ring R, then it is clear by using the projection map π_i that any solution for R implies a solution for SIP-LWE in R_i with induced secret key and error distributions $\chi_s^{(i)} = \chi_s(R) \bmod f_i(x)$ and $\chi_e^{(i)} = \chi_e(R) \bmod f_i(x)$. However, the converse is also true: any solution to SIP-LWE in R_i can immediately be lifted to a solution for R. It suffices to take the vector $\boldsymbol{a}_*$ (resp. $\boldsymbol{b}$) to be zero on all components $R_j \neq R_i$, and equal to the solution found for R_i in component i.

We can thus repeat the above three approaches but now for R_i and the induced secret key and error distributions $\chi_s^{(i)}$ and $\chi_e^{(i)}$, which shows that

$$\mathtt{Adv}^{\text{SIP-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{A}) \geq \max \left\{ \begin{array}{l} \mathtt{Adv}^{\text{Search-Mod-LWE}}_{R,l,\chi_s^{(i)},\chi_e^{(i)}}(\mathcal{B}), \\ 2^{-H_\infty(\chi_s^{(i)}(R))}, \\ 2^{-H_\infty(\chi_e^{(i)}(R))} \end{array} \right\}_{i=1,\ldots,t} . \tag{1}$$

Remark 1. Note that in general the distributions $\chi_s^{(i)}$ (and $\chi_e^{(i)}$) can be quite intricate, and if $|R_i|$ is small compared to $|R|$ both will be indistinguishable from uniform random as shown in [Pei16, Theorem 5.2]. In this case Search-Mod-LWE$_{R,l,\chi_s^{(i)},\chi_e^{(i)}}$ essentially reduces to guessing the secret part in R_i, and thus is the same as the first approach. In all cases, we have that the min entropy of both is upper bounded by the min entropy of the uniform distribution on R_i, i.e. $H_\infty(\chi_*^{(i)}) \leq \deg(f_i) \cdot \log_2 q$, for $* = e$ or s. This shows that solving SIP-LWE is easy for rings with q and $\deg(f_i)$ small enough, simply by solving it in a factor R_i (corresponding to guessing the answer in R_i).

An important example are rings R of the form $\mathbb{Z}_q[X]/(X^N+1)$ with q an NTT-friendly prime, in particular, X^N+1 splits completely modulo q in linear factors. Using the Chinese remainder theorem, we see that $R \cong R_1 \times \cdots \times R_N$ with $R_i = \mathbb{Z}_q[X]/(X-\alpha_i)$ and α_i the roots of X^N+1 modulo q. Since $\deg(f_i) = 1$, we therefore conclude that the min entropy of $\chi_s^{(i)}$ and $\chi_e^{(i)}$ are upper bounded by $\log_2 q$, which is tiny compared to the min entropy of $\chi_s(R)$ and $\chi_e(R)$ itself. Again this shows that SIP-LWE is easy for such rings.

Remark 2. Instead of using the smallest factors R_i as done above, it is also possible to reduce the SIP-LWE problem modulo larger factors, i.e. to reduce modulo any divisor $g(x) | f(x) \bmod q$. For the first two approaches, i.e. guessing part of the secret or error, the goal is to get the entropy of $\chi_*(R) \bmod g$ as small as possible, so it does not make sense to use larger factors. Furthermore, for properly instantiated Mod-LWE problems as described in [Pei16], reducing the Search-Mod-LWE problem into a smaller factor does not make the problem easier, so for the third approach it does not make sense to do so.

Hard instances of SIP-LWE. We have showed that SIP-LWE can only be hard for rings R that have no small factors (including the ring R itself) and when Search-Mod-LWE is hard for R (which implied hardness when reduced into smaller factors).

Due to the existence of easy instances that depend on how $f(x)$ splits modulo q, it is tempting to apply modulus switching and solve the problem modulo some q' such that $f(x)$ splits completely modulo q' for instance, and then lift the solution back to the original q. To see that this approach cannot work in general, it suffices to consider the rank one case: assume the orginal q is such that R is a field, then we know that SIP-LWE in this case is at least as hard as Search-Mod-LWE, so if modulus switching would work, it immediately results in an attack on Search-Mod-LWE (which we assumed to be hard).

Although we have no proof (except for the rank one case) that the lower bound in (1) is in fact tight, we will assume this to be the case for the instances we consider in the remainder of the paper. In particular, we will only apply our transformation to instances of low rank and where the ring R has no small factors, e.g. R a field. Determining the exact hardness of SIP-LWE however remains an open problem.

A discussion of the hardness of SIP(0)-LWE of several practical schemes is given in subsection 5.2.

4 Error Term Checking

Here we describe our two new chosen ciphertext transformations and relate their security to the hardness of the SIP-LWE and SIP0-LWE problems.

4.1 Error Term Checking 1

Using the fact that $\boldsymbol{v}'_d - \boldsymbol{v} = \boldsymbol{s}'^T\boldsymbol{e} - \boldsymbol{e}'^T\boldsymbol{s} + \boldsymbol{e}'' + \boldsymbol{u} + \mathtt{encode}(m)$, we propose a new transformation to transform an OW-CPA secure PKE into an IND-CCA secure KEM. The $\mathsf{KEM}_{\mathtt{ETC1}}.\mathtt{KeyGen}$ and $\mathsf{KEM}_{\mathtt{ETC1}}.\mathtt{Encaps}$ function are identical to the functions in the original $\mathsf{KEM}_{\mathtt{FO}}$. During the decapsulation the input $\boldsymbol{v}'_c$ is completely recalculated and checked as in the original FO transformation, but $\boldsymbol{b}'$ is not recalculated and is instead checked using the error term equation. The new decapsulation function is given in Figure 3.

The security of this transformation can be bound similarly to the FO transformation, using an additional term that involves the hardness of the SIP0-LWE problem. Furthermore, the γ spread of $\mathsf{PKE}.\mathsf{Enc}$ is replaced by the γ spread of the $\mathsf{KEM}_{\mathtt{ETC1}}.\mathtt{GenET}$, which can be lower bounded by the min-entropy of the generation $\chi_e(R_q^{1\times 1};\cdot)$ using Proposition 1. This lower bound is typically high enough so that the term is negligible in comparison to the other terms. The main idea behind the proof is to follow the proof of the FO transformation, with an additional game where the difference between the $\mathtt{FO}$ transformation and $\mathtt{ETC1}$ transformation is bound. The full proof is given in Appendix A. A sketch of the proof in the QROM model, which uses an analogous adaptation from the FO proof in the quantum random oracle model, can be found in Appendix C.

$\mathsf{KEM}_{\mathtt{ETC1}}.\mathtt{Decaps}(sk=(\boldsymbol{s},\boldsymbol{e},w),$ $pk=(\boldsymbol{b},\mathrm{seed}_{\boldsymbol{A}}),c=(\boldsymbol{v}'_c,\boldsymbol{b}'))$	$\mathsf{KEM}_{\mathtt{ETC1}}.\mathtt{GenET}(sk=(\boldsymbol{s},\boldsymbol{e}),$ $pk=(\boldsymbol{b},\mathrm{seed}_{\boldsymbol{A}}),\overline{m};\overline{r})$
1. $\overline{m}:=\mathsf{PKE.Dec}(sk,c)$) // decrypt 2. $\overline{r}:=\mathcal{G}(\overline{m},pk)$ 3. $(\overline{\boldsymbol{v}},\overline{\boldsymbol{v}'_c}):=\mathsf{KEM}_{\mathtt{ETC1}}.\mathtt{GenET}(sk,pk,\overline{m};\overline{r})$ 4. **if** $\boldsymbol{b}'^T\boldsymbol{s}=\overline{\boldsymbol{v}}$ and $\boldsymbol{v}'_c=\overline{\boldsymbol{v}'_c}$ 5. **then return** $K:=\mathcal{H}(\overline{m},c)$ 6. **else return** $K:=\mathcal{H}(w,c)$	1. $\overline{\boldsymbol{s}'}\leftarrow\chi_s(R_q^{1\times l};\overline{r})$ 2. $\overline{\boldsymbol{e}'}\leftarrow\chi_e(R_q^{1\times l};\overline{r})$ 3. $\overline{\boldsymbol{e}''}\leftarrow\chi_e(R_q^{1\times 1};\overline{r})$ 4. $\overline{\boldsymbol{v}'}=\boldsymbol{b}^T\overline{\boldsymbol{s}'}$ 5. $\overline{\boldsymbol{v}'_m}=\overline{\boldsymbol{v}'}+\overline{\boldsymbol{e}''}+\mathtt{encode}(\overline{m})$ 6. $\overline{\boldsymbol{v}'_c}:=\mathtt{compress}(\overline{\boldsymbol{v}'_m})$ 7. $\overline{\boldsymbol{z}}:=\overline{\boldsymbol{s}'}\boldsymbol{e}-\overline{\boldsymbol{e}'}\boldsymbol{s}+\overline{\boldsymbol{e}''}+\mathtt{encode}(\overline{m})$ 8. $\overline{\boldsymbol{v}}:=\overline{\boldsymbol{v}'_m}-\overline{\boldsymbol{z}}$ 9. **return** $(\overline{\boldsymbol{v}},\overline{\boldsymbol{v}'_c})$

Fig. 3: IND-CCA secure $\mathsf{KEM}_{\mathtt{ETC1}}$

Theorem 2. *Given a δ-correct PKE, for any classical adversary $\mathcal{A}$ against the IND-CCA security of $\mathsf{KEM}_{\mathtt{ETC1}}$ issuing at most q_D queries to the decapsulation oracle and at most q_G and q_H queries to the random oracles $\mathcal{G}$ and $\mathcal{H}$ respectively, there exists a classical adversary $\mathcal{B}$ against the SIP0-LWE problem and a classical adversary $\mathcal{C}$ against the OW-CPA security of the underlying encryption scheme such that:*

$$\mathtt{Adv}^{ind\text{-}cca}_{ETC1}(\mathcal{A})\leq q_H/2^{l_m}+q_D2^{-\gamma}+q_G\delta+q_Dq_G\mathtt{Adv}^{SIP0\text{-}LWE}_{R,l,\chi_s,\chi_e}(\mathcal{B})+(q_G+q_H)\mathtt{Adv}^{ow\text{-}cpa}_{pke}(\mathcal{C}),$$

where γ is the spread of the function $\mathsf{KEM}_{\mathtt{ETC1}}.\mathtt{GenET}()$, and where the running time of $\mathcal{A}$ is not much higher than the running time of $\mathcal{C}$ or $\mathcal{B}$.

4.2 Error Term Checking 2

For schemes where compression of $\boldsymbol{v}'_m$ during encryption is not lossy, i.e. $\boldsymbol{u}=0$, it is possible to improve the efficiency of the ETC1 transformation. We will denote this transformation with ETC2. During $\mathsf{KEM}_{\mathtt{ETC2}}.\mathtt{Decaps}$ we only check the validity of the error term, and do not recompute the values $\overline{\boldsymbol{v}'_c}$ or $\overline{\boldsymbol{v}'}$. This makes the implementation more efficient than the ETC1 transformation. The security of the new transformation can be bounded by the SIP-LWE problem. The proof of this theorem proceeds analogous to the proof of the ETC1 transformation and is given in Appendix B. A sketch of the proof in the QROM model can be found in Appendix C.

$\mathsf{KEM}_{\mathtt{ETC2}}.\mathtt{Decaps}(sk=(\boldsymbol{s},\boldsymbol{e},w),$ $pk=(\boldsymbol{b},\mathrm{seed}_{\boldsymbol{A}}),c=(\boldsymbol{v}'_c,\boldsymbol{b}'))$	$\mathsf{KEM}_{\mathtt{ETC2}}.\mathtt{GenET}(sk=(\boldsymbol{s},\boldsymbol{e}),$ $pk=(\boldsymbol{b},\mathrm{seed}_{\boldsymbol{A}}),\overline{m};\overline{r})$
1. $\overline{m}:=\mathsf{PKE.}(sk,c)$) // decrypt 2. $\overline{r}:=\mathcal{G}(\overline{m},pk)$ 3. $\overline{\Delta\boldsymbol{v}}:=\mathsf{KEM}_{\mathtt{ETC2}}.\mathtt{GenET}(sk,pk,\overline{m};\overline{r})$ 4. **if** $\boldsymbol{v}'-\boldsymbol{b}'^T\boldsymbol{s}==\overline{\Delta\boldsymbol{v}}$ 5. **then return** $K:=\mathcal{H}(\overline{m},c)$ 6. **else return** $K:=\mathcal{H}(w,c)$	1. $\overline{\boldsymbol{s}'}\leftarrow\chi_s(R_q^{1\times l};\overline{r})$ 2. $\overline{\boldsymbol{e}'}\leftarrow\chi_e(R_q^{1\times l};\overline{r})$ 3. $\overline{\boldsymbol{e}''}\leftarrow\chi_e(R_q^{1\times 1};\overline{r})$ 4. $\overline{\boldsymbol{z}}:=\overline{\boldsymbol{s}'}\boldsymbol{e}-\overline{\boldsymbol{e}'}\boldsymbol{s}+\overline{\boldsymbol{e}''}$ 5. $\overline{\Delta\boldsymbol{v}}:=\mathtt{encode}(\overline{\boldsymbol{z}},m)$ 6. **return** $\overline{\Delta\boldsymbol{v}}$

Fig. 4: IND-CCA secure $\mathsf{KEM}_{\mathtt{ETC2}}$

Theorem 3. *For any classical adversary $\mathcal{A}$ against the IND-CCA security of $\mathsf{KEM}_{\mathtt{ETC2}}$ issuing at most q_D queries to the decryption oracle and at most q_G and q_H queries to the random oracles $\mathcal{G}$ and $\mathcal{H}$ respectively, there exists a classical adversary $\mathcal{B}$ against the SIP-LWE problem and a classical adversary $\mathcal{C}$ against the IND-CPA security of the underlying encryption scheme such that:*

$$Adv^{ind\text{-}cca}_{ETC2}(\mathcal{A}) \leq q_H/2^{l_m} + q_D 2^{-\gamma} + q_G\delta + q_D q_G Adv^{SIP\text{-}LWE}_{R,l,\chi_s,\chi_e}(\mathcal{B}) + (q_G+q_H)Adv^{ow\text{-}cpa}_{pke}(\mathcal{C})$$

where γ is the spread of the function $\mathsf{KEM}_{\mathtt{ETC2}}.\mathtt{GenET}()$, and where the running time of $\mathcal{A}$ is not much higher than the running time $\mathcal{C}$ or $\mathcal{B}$.

5 Applications

In this section we will compare the efficiency of $\mathtt{KEM}_{\mathtt{ETC1}}$ and $\mathtt{KEM}_{\mathtt{ETC2}}$ with the original $\mathtt{KEM}_{\mathtt{FO}}$. We will first make an overall comparison between the KEMs and discuss the applicability of $\mathtt{KEM}_{\mathtt{ETC1}}$ and $\mathtt{KEM}_{\mathtt{ETC2}}$ on several NIST post-quantum standardization candidates, after which we apply the ETC2 transformation to the NIST post-quantum standardization candidate Threebears.

5.1 Comparison to FO

In any of the three transformations, the PKE decryption is performed first, after which the secret terms $\overline{\boldsymbol{s}'}$,$\overline{\boldsymbol{e}'}$,$\overline{\boldsymbol{e}''}$ are recalculated as in the encryption.

$\mathsf{KEM}_{\mathtt{FO}}$ proceeds by recalculating both $\boldsymbol{b}'$ and $\boldsymbol{v}'_c$. Recalculating $\boldsymbol{b}'$ requires the regeneration of the matrix $\boldsymbol{A}$ and l^2 multiplications of elements in R, while the recalculation of $\boldsymbol{v}'$ costs l multiplications of elements in R. Note that the generation of $\boldsymbol{A}$ and polynomial multiplications are typically the most costly operations in lattice-based encryption schemes.

$\mathsf{KEM}_{\mathtt{ETC1}}$ does recalculate $\boldsymbol{v}'_c$ but does no longer calculate $\boldsymbol{b}'$. Thus it gets rid of the regeneration of $\boldsymbol{A}$ and the l^2 multiplications of elements in R. Instead it requires $2{\cdot}l$ multiplications of elements in R, with the difference that the newly introduced multiplications use only small elements as generated by the distributions χ_s and χ_e, a fact that could be used to further speed up the process using specific multiplication algorithms that require small coefficients (e.g. [CHK+20]), or using the fact that modular reductions could become obsolete.

$\mathsf{KEM}_{\mathtt{ETC2}}$ gets rid of both the calculation of $\boldsymbol{b}'$ and $\boldsymbol{v}'_c$, and replaces them with $2{\cdot}l$ multiplications of small elements in R. Table 1 gives an overview of the main operations in PKE.Dec and the additional operations introduced by following the FO and the ETC1 transformation respectively.

5.2 Applicability on NIST submissions

To be able to apply the ETC1 or ETC2 transformation, the underlying encryption scheme needs to meet specific criteria: the relevant SIP-LWE problem needs to be a hard problem and the encryption scheme should fit the description as given

	PKE.Dec	$\mathsf{KEM}_{\mathtt{FO}}$	$\mathsf{KEM}_{\mathtt{ETC1}}$	$\mathsf{KEM}_{\mathtt{ETC2}}$
$\mathtt{gen}_A(seed_A)$	0	1	0	0
mult	l	$(l+1)\cdot l$	$3\cdot l$	$2\cdot l$
▶$\mathtt{mult}_{\mathrm{big}}$	l	$(l+1)\cdot l$	$1\cdot l$	0
▶$\mathtt{mult}_{\mathrm{small}}$	0	0	$2\cdot l$	$2\cdot l$

Table 1: Operations needed for PKE.Dec and additional operations needed for FO, ETC1 and ETC2 transformations.

in Figure 1. Specifically, the ETC1 transformation requires that the SIP0-LWE problem with parameters R,l,χ_s and χ_e similar to the encryption scheme is hard. Moreover, there should not be any compression of the public key element $\boldsymbol{b}$ or the ciphertext element $\boldsymbol{b'}$. The conditions for applying ETC2 are more strict: the relevant SIP-LWE problem should be hard and there should be no compression of the public key $\boldsymbol{b}$ or ciphertext elements $\boldsymbol{b'}$ and $\boldsymbol{v'}$.

The hardness of the SIP-LWE problem is discussed in subsection 3.1. For schemes where R is a field, SIP-LWE and SIP0-LWE are hard if the distribution χ_s has enough min-entropy. This is the case for Threebears [Ham19]. If R splits in multiple subrings the entropy of $\chi_s^{(i)}$ and $\chi_e^{(i)}$ is a measure for the hardness of the SIP-LWE problem. As remarked in subsection 3.1, this implies that schemes that use NTT rings, such as NewHope [PAA+19] or Kyber [BDK+18] are typically not fit for usage of the ETC1 or ETC2 transformations. For Saber [DKRV18] the entropy in the subrings is also too low for SIP-LWE to be secure. In case of LAC [LLJ+19], the ring $\mathbb{Z}_{251}[X]/(X^{1024}+1)$ splits in only 2 subrings which should keep the entropy of χ_s in the subrings high enough to make the SIP-LWE problem hard.

An exceptional case occurs when $l=1$, in which case the SIP0-LWE problem is unsolvable if $\boldsymbol{s}$ is invertible as discussed in subsection 3.1. This means that the ETC1 transformation can be applied to NewHope if during key encapsulation $\boldsymbol{s}$ is forced to be invertible. Experimentally, we observed that the probability of a randomly generated $\boldsymbol{s}$ being invertible is 91.4%, by generating 2^{14} polynomials $\boldsymbol{s}$ and checking them for invertibility.

Public key and ciphertext compression is relatively common in lattice-based schemes. All discussed schemes do compression of the ciphertext term $\boldsymbol{v'}$ and are thus not eligible to use in the ETC2 transformation. However, the first version of LAC did only a slight compression of the $\boldsymbol{v'}$ in which one of the coefficients of $\boldsymbol{v'}$ was not communicated. In this case, removing this compression would be enough to allow usage of ETC2. Interesting future work could be to look at adapting the ETC2 transformation to schemes that use compression.

5.3 Case study: Threebears

The Threebears cryptosystem by Hamburg [Ham19] is a Post-Quantum KEM accepted to the second round of the NIST Post-Quantum Cryptography Stan-

	R	l	ETC1	ETC2	SIP0-LWE hard?	SIP-LWE hard?
Threebears	$\mathbb{Z}_{2^{3120}-2^{1560}-1}$	2-4	✓	✗	✓	✓
LAC	$\mathbb{Z}_{251}[X]/(X^{1024}+1)$	1	✓	(✓)†	✓	✓
NewHope	$\mathbb{Z}_{12289}[X]/(X^{1024}+1)$	1	(✓)*	✗	(✓)*	✗

† For LAC without compression.
* If non invertible secrets are rejected.

Table 2: Overview of applicability of ETC1 and ETC2 transformation on selection of NIST round 2 candidates.

dardization process. It consists of three schemes: BabyBear, MamaBear and PapaBear, listed in order of increasing security.

The security of Threebears is based on the Integer-LWE problem [Chu17], which is a mathematical hard problem that is similar to the Mod-LWE problem. In Threebears the ring R is isomorphic to $\mathbb{Z}_N$, where N is the prime $2^{3120}-2^{1560}-1$, and l is a value ranging from 2 to 4 depending on the security level. The secrets are generated by sampling polynomials in the ring $\mathbb{Z}_q[X]/(X^{312}-X^{156}-1)$ with coefficients according to a distribution with small variance. These polynomials are then converted to an element of R by substituting $X=q=2^{10}$. For a full description of Threebears we refer to the NIST submission package [Ham19].

We implemented[1] a $\texttt{KEM}_{\texttt{ETC1}}$ version of ThreeBears based on the original code provided by Hamburg[2] starting from the 'opt_cache' optimized version of ThreeBears, which caches the full secret key. In our implementation, `KeyGen` is changed only slightly to save $\boldsymbol{e}$, `Encaps` is not changed and `Decaps` is rewritten following the $\texttt{KEM}_{\texttt{ETC1}}$ specifications. We did not optimize the multiplications procedure to account for the fact that some multiplications are between small elements in R, which could boast an interesting future speedup.

In Table 3 we compare the $\texttt{KEM}_{\texttt{FO}}$ implementation with our $\texttt{KEM}_{\texttt{ETC1}}$ implementation. The tests were performed on an Intel(R) Core(TM) i7-10510U processor. We can see that the key generation takes slightly longer for ETC1 compared with FO, which is due to the additional operations to save $\boldsymbol{e}$. The encryption phase remains the same in both transformations, which is reflected in the timings. The main speedup comes during the decapsulation, where we can observe a significant difference in timing. We observe a 12.7%,29.7% and 37,4% speedup of the full decapsulation for BabyBear, MamaBear and PapaBear respectively. This is in line with the results of table 1 as l increases for higher security levels.

[1] The code can be found at *anonymised*
[2] https://sourceforge.net/projects/threebears/; commit [f4ce0e]

	BabyBear		MamaBear		PapaBear	
	FO	ETC1	FO	ETC1	FO	ETC1
KeyGen	25	25	51	53	73	76
Encaps	36	35	59	58	88	88
Decaps	47	**41**	74	**52**	107	**67**

Table 3: Timing (μs) for the different versions of ThreeBears.

6 Conclusion and Future Work

In this paper we showed that it is possible to design specific chosen ciphertext transformations for lattice based encryption schemes that outperform generic transformations. We presented two new transformations to compile an IND-CPA secure encryption scheme into an IND-CCA secure KEM. As opposed to generic transformations such as FO, our transformations, ETC1 and ETC2, are specifically designed with lattice based encryption schemes in mind. These new transformations change the way ciphertexts are checked during decapsulation, removing the need for the reconstruction of the public matrix $\boldsymbol{A}$ and changing the nature and number of polynomial multiplications. For the NIST Round 2 candidate Threebears we have shown that our method speeds up the decapsulation with up to 37.4% for the highest security parameters. Importantly, ETC1 and ETC2 do not change the communication between both parties and can thus be used without changing the specifications of already designed algorithms.

At the moment, the conditions to apply the ETC1 and ETC2 transformations are strict, which means that they can only be employed to a subset of lattice based schemes. It would be interesting future work to look into expanding the support of these specific transformations to a wider set of lattice based schemes.

Acknowledgements

The authors would like to thank Alice Pellet–Mary, Léo Ducas and Chris Peikert for their valuable insights into the security of the SIP-LWE problem.

This work was supported in part by CyberSecurity Research Flanders with reference number VR20192203, the Research Council KU Leuven grants C16/15/058 and C14/18/067, the SRC grant 2909.001, the Horizon 2020 ERC Advanced Grant (695305 Cathedral) and the ERC Advanced Grant ERC-2015-AdG-IMPaCT.

References

AHU19. Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In *CRYPTO 2019, Part II*, pages 269–295, 2019.

AOP+17. Martin R. Albrecht, Emmanuela Orsini, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. Tightly secure ring-LWE based key encapsulation with short ciphertexts. In *ESORICS 2017, Part I*, pages 29–46, 2017.

ARU14. Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *55th FOCS*, pages 474–483, 2014.

BDK+18. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle. Crystals - kyber: A cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 353–367, 2018.

BHH+19. Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In *TCC 2019, Part II*, pages 61–90, 2019.

BR95. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT'94*, 1995.

CHK+20. Chi-Ming Marvin Chung, Vincent Hwang, Matthias J. Kannwischer, Gregor Seiler, Cheng-Jhih Shih, and Bo-Yin Yang. Ntt multiplication for ntt-unfriendly rings. Cryptology ePrint Archive, Report 2020/1397, 2020. https://eprint.iacr.org/2020/1397.

Chu17. Gu Chunsheng. Integer version of ring-LWE and its applications. Cryptology ePrint Archive, Report 2017/641, 2017. http://eprint.iacr.org/2017/641.

Den03. Alexander W. Dent. A designer's guide to KEMs. In *9th IMA International Conference on Cryptography and Coding*, pages 133–151, 2003.

DKRV18. Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In *AFRICACRYPT 18*, pages 282–305, 2018.

Flu16. Scott Fluhrer. Cryptanalysis of ring-LWE based key exchange with key share reuse. Cryptology ePrint Archive, Report 2016/085, 2016. http://eprint.iacr.org/2016/085.

FO13. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, (1):80–101, 2013.

GN07. Nicolas Gama and Phong Q. Nguyen. New chosen-ciphertext attacks on NTRU. In *PKC 2007*, pages 89–106, 2007.

Ham19. Mike Hamburg. Three Bears. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

HHK17. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In *TCC 2017, Part I*, pages 341–371, 2017.

HNP+03. Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In *CRYPTO 2003*, pages 226–246, 2003.

JJ00. Éliane Jaulmes and Antoine Joux. A chosen-ciphertext attack against NTRU. In *CRYPTO 2000*, pages 20–35, 2000.

JZC+18. Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In *CRYPTO 2018, Part III*, pages 96–125, 2018.

JZM19. Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model. In *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 227–248, 2019.

KSS+20. Veronika Kuchta, Amin Sakzad, Damien Stehlé, Ron Steinfeld, and Shifeng Sun. Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 703–728. Springer, 2020.

LLJ+19. Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, pages 1–23, 2010.

NC11. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.

NIS16. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process, 2016.

PAA+19. Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

Pei16. Chris Peikert. How (not) to instantiate ring-lwe. In *SCN*, volume 9841 of *Lecture Notes in Computer Science*, pages 411–430. Springer, 2016.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84–93, 2005.

Sho01. Victor Shoup. Oaep reconsidered. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, page 239–259, Berlin, Heidelberg, 2001. Springer-Verlag.

SSTX09. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT 2009*, pages 617–635, 2009.

SXY18. Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In *EUROCRYPT 2018, Part III*, pages 520–551, 2018.

TU16. Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In *TCC 2016-B, Part II*, pages 192–216, 2016.

Unr14. Dominique Unruh. Revocable quantum timed-release encryption. In *EUROCRYPT 2014*, pages 129–146, 2014.

ZCH+19. Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, William Whyte, John M. Schanck, Andreas Hulsing, Joost Rijneveld, Peter Schwabe, and Oussama Danba. NTRUEncrypt. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

Zha12. Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In *CRYPTO 2012*, pages 758–775, 2012.

A Security proof of Theorem 2

Proof. The main idea of the proof is to transform the IND-CCA game for the ETC1 transform into the IND-CCA game of the FO transform using a game hopping technique. An overview of the games can be found in Figure 5.

Game G_0: Let $\mathcal{A}$ be an adversary against the IND-CCA security of the KEM constructed using the ETC1 transformation, $\texttt{KEM}_{\texttt{ETC1}}$. Then by definition, the first game G_0 is the same as the IND-CCA security game of $\texttt{KEM}_{\texttt{ETC1}}$, or:

$$|\Pr[G_0^{\mathcal{A}}=1]-\frac{1}{2}|=\text{Adv}^{\text{ind-cca}}_{\text{ETC1}}(\mathcal{A}). \tag{2}$$

Game G_1: In game G_1 we introduce the oracle $\mathcal{H}'$ which is hidden for the adversary and which replaces a query $\mathcal{H}(w,.)$ during the decapsulation. The oracle $\mathcal{H}$ is changed so that on a query where $m=w$ the QUERY_H flag is raised and the game aborts. These changes remain unnoticed until the adversary queries $\mathcal{H}(w,.)$. As w is drawn from the uniform distribution over strings with l_m bits, and considering a maximum of q_H queries to the oracle $\mathcal{H}$, we get:

$$|\Pr[G_1=1]-\Pr[G_0=1]|\leq q_H/2^{l_m}. \tag{3}$$

Game G_2: In Game G_2, the validity check in the decapsulation oracle is performed by checking previous queries to the random oracle $\mathcal{G}$. It accepts the check if there exists a previous query $(\overline{m},r)$ to the random oracle such that $(\boldsymbol{b}'\boldsymbol{s},\boldsymbol{v}'_c)=\texttt{KEM}_{\texttt{ETC1}}.\texttt{GenET}(sk,pk,\overline{m};r)$.

If the ciphertext is accepted in G_2, then $\mathcal{G}$ was queried in $\overline{m}$ returning $\overline{r}$ and together with the fact that:

$$(\boldsymbol{b}'\boldsymbol{s},\boldsymbol{v}'_c)=\texttt{KEM}_{\texttt{ETC1}}.\texttt{GenET}(sk,pk,\overline{m};\overline{r}),$$

we know that the ciphertext will also be accepted in G_1. If the ciphertext is accepted in G_2, then it is not accepted in G_1 if and only if the random oracle $\mathcal{G}$ was not queried in $\overline{m}$. As the ciphertext still has to fulfill $(\boldsymbol{b}'\boldsymbol{s},\boldsymbol{v}'_c)=\texttt{KEM}_{\texttt{ETC1}}.\texttt{GenET}(sk,pk,\overline{m};\mathcal{G}(\overline{m}))$, the probability of finding a difference between both games is smaller than $2^{-\gamma}$, with γ the spread of the function $\texttt{KEM}_{\texttt{ETC1}}.\texttt{GenET}(sk,pk,m;r)$. From this it follows that:

$$|\Pr[G_2=1]-\Pr[G_1=1]|\leq q_D 2^{-\gamma}, \tag{4}$$

with q_D the number of decapsulation oracle queries.

Game G_3: In Game G_3, we don't check if $m=\overline{m}$ anymore. Remember that during decapsulation the message is calculated as $\overline{m}:=\texttt{decode}(\boldsymbol{v}'_d-\boldsymbol{v})$, and due to the checks we know that:

$$\boldsymbol{v}=\boldsymbol{b}'^T\boldsymbol{s}=\overline{\boldsymbol{v}'_m}-\overline{\boldsymbol{z}} \text{ and } \boldsymbol{v}'_d=\texttt{decompress}(\texttt{compress}(\overline{\boldsymbol{v}'_m})),$$

GAME G_0-G_3.

1. $(pk,sk) \leftarrow \texttt{KeyGen}()$
2. $w \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
3. $m^* \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
4. $r^* := \mathcal{G}(m^*)$ // G_0-G_4
5. $r^* := \mathcal{U}(\{0,1\}^{l_r})$ // G_5
6. $c^* := \texttt{PKE.Enc}(pk,m^*;r^*)$
7. $k_0 := \mathcal{H}(m^*,c^*)$ // G_0-G_4
8. $k_0 \leftarrow \mathcal{K}$ // G_5
9. $k_1 \leftarrow \mathcal{K}$
10. $b \leftarrow \mathcal{U}(\{0,1\})$
11. $b' \leftarrow A^{\texttt{Decaps},\mathcal{H},\mathcal{G}}(pk,c^*,k_b)$
12. return $b = b'$

$\mathcal{H}(m,c)$:

1. if: $\exists K : (m,c,K) \in \mathcal{L}_H$
2. return K
3. $K \leftarrow \mathcal{K}$
4. if: $m = w$ // G_1-G_5
5. QUERY$_H$:= true // G_1-G_5
6. **abort** // G_1-G_5
7. if: $m = m^*$ and $c = c^*$ // G_5
8. CHAL := true // G_5
9. **abort** // G_5
10. $\mathcal{L}_H := \mathcal{L}_H \cup (m,c,K)$
11. return K

$\mathsf{Decaps}(sk = (\boldsymbol{s},\boldsymbol{e}), pk = (\boldsymbol{b},\text{seed}_{\boldsymbol{A}}), c = (\boldsymbol{v}'_c,\boldsymbol{b}'))$:

1. $t := (\boldsymbol{b}'\boldsymbol{s},\boldsymbol{v}'_c)$ // G_0-G_3
2. $\overline{m} := \texttt{PKE.Dec}(sk,c)$ // G_0-G_2
3. $\overline{r} := \mathcal{G}(\overline{m},pk)$ // $G_0 - G_1$
4. if: $t = \texttt{KEM}_{\text{ETC1}}.\texttt{GenET}(sk,pk,\overline{m};\overline{r})$ // $G_0 - G_1$
5. if: $\exists (m,r) \in \mathcal{L}_G$ and $m = \overline{m}$ and $t = \texttt{KEM}_{\text{ETC1}}.\texttt{GenET}(sk,pk,m;r)$// G_2
6. if: $\exists (m,r) \in \mathcal{L}_G$ and $t = \texttt{KEM}_{\text{ETC1}}.\texttt{GenET}(sk,pk,m;r)$ // G_3
7. if: $\exists (m,r) \in \mathcal{L}_G : c = \texttt{PKE.Enc}(pk,m,r)$ // G_4
8. if: $m = w$ // G_1-G_5
9. return $K := \mathcal{H}'(c)$ // G_1-G_5
10. return $K := \mathcal{H}(m,c)$
11. else:
12. return $K := \mathcal{H}(w,c)$ // G_0
13. return $K := \mathcal{H}'(c)$ // G_1-G_5

$\mathcal{G}(m)$:

1. if: $\exists r : (m,r) \in \mathcal{L}_G$
2. return r
3. if: $m = m^*$ // G_5
4. CHAL$_G$:= true // G_5
5. **abort** // G_5
6. $r \leftarrow \mathcal{U}(\{0,1\}^{l_r})$
7. $\mathcal{L}_G := \mathcal{L}_G \cup (m,r)$
8. return r

Fig. 5: Games G_0 to G_6 as used in the proof of theorem 2

and thus we can write difference $\boldsymbol{v}'_d - \boldsymbol{v}$ as:

$$\overline{\boldsymbol{s}'}\boldsymbol{e} - \overline{\boldsymbol{e}'}\boldsymbol{s} + \overline{\boldsymbol{e}''} + \overline{\boldsymbol{u}} + \texttt{encode}(m).$$

This is exactly the error condition as as discussed in section 2.3. The probability of failure is defined as δ, and thus finding a failing ciphertext using q_G queries to the oracle $\mathcal{G}$ has a probability bound by $q_G\delta$, which results in:

$$|\Pr[G_3 = 1] - \Pr[G_2 = 1]| \leq q_G\delta. \tag{5}$$

Game G_4: In Game G_4, the validity check using the error term is replaced by the validity check of the FO transformation.

If a check in game G_4 is accepted, then c was correctly generated and therefore the check in game G_3 will be accepted. The only way to differentiate between both games is thus finding a ciphertext that is accepted in game G_3 and rejected in game G_4, and thus we can say that distinguishing both games is as least as hard as submitting a ciphertext query that differentiates both games.

We construct an adversary $\mathcal{B}$ that uses the adversary $\mathcal{A}$ capable of generating such a differentiating query to solve the SIP0-LWE problem. The decapsulation oracle is constructed as in Figure 6. As long as there is no differentiating query, the `Decaps` function as given in Figure 6 perfectly mimics the `Decaps` in both G_3 and G_4. If $\mathcal{A}$ submits a differentiating ciphertext c_{diff} that is resolved differently in both games, we keep on replying with the same decapsulation algorithm to the adversary $\mathcal{A}$ until he returns a value b', even though the calculations from this point are no longer relevant for $\mathcal{B}$ as the differentiating query is already in the list $\mathcal{L}_D$. Note that $\mathcal{B}$ doesn't know when a differentiating ciphertext has been queried.

If the adversary was able to differentiate between both games, he queried a differentiating ciphertext and with a probability of at least $1/q_D$, the ciphertext $c = (\hat{\boldsymbol{v}}'_c, \hat{\boldsymbol{b}}')$ drawn from the decapsulation list $\mathcal{L}_D$ in line 9 is this differentiating ciphertext c_{diff}. If this happens there is a probability $1/q_G$ that (m', r') drawn in line 10 is the pair (m', r') so that:

$$(\boldsymbol{b}'\boldsymbol{s}, \boldsymbol{v}'_c) = \texttt{KEM}_{\texttt{ETC1}}.\texttt{GenET}(sk, pk, m'; r').$$

This means that:

$$(\hat{\boldsymbol{v}}'_c, \hat{\boldsymbol{b}}') \neq \texttt{PKE.Enc}(pk, m'; r'),$$

due to the check rejection in G_4 and:

$$\boldsymbol{v}'_c = \hat{\boldsymbol{v}}'_c \text{ and } \boldsymbol{b}'\boldsymbol{s} = \boldsymbol{v}'_m - \boldsymbol{z} = \hat{\boldsymbol{b}}'\boldsymbol{s},$$

due to the acceptance in G_3, where $(\boldsymbol{b}', \boldsymbol{v}'_c) := \texttt{PKE.Enc}(pk, m'; r')$. From this we can say that $\boldsymbol{b}' \neq \hat{\boldsymbol{b}}'$ and that $(\boldsymbol{b}' - \hat{\boldsymbol{b}}')\boldsymbol{s} = 0$, thus solving the SIP0-LWE problem.

Overall this means that if $\mathcal{A}$ can distinguish between the games, $\mathcal{B}$ can construct a solution to the SIP0-LWE problem with probability $1/q_D q_G$ and therefore:

$$|\Pr[G_4 = 1] - \Pr[G_3 = 1]| \leq q_D q_G \text{Adv}^{\text{SIP0-LWE}}_{R,l,\chi_s,\chi_e}(\mathcal{B}).$$

$\mathcal{B}(pk=(\boldsymbol{b},seed_{\boldsymbol{A}}))$:

1. $m^* \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
2. $r^* := \mathcal{G}(m^*)$
3. $c^* := \texttt{PKE.Enc}(pk,m^*;r^*)$
4. $k_0 := \mathcal{H}(m^*,c^*)$
5. $k_1 \leftarrow \mathcal{K}$
6. $b \leftarrow \mathcal{U}(\{0,1\})$
7. $w \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
8. $b' \leftarrow \mathcal{A}^{\texttt{Decaps}(pk,\cdot),\mathcal{H},\mathcal{G}}(pk,c^*,k_b)$
9. $\hat{\boldsymbol{v}}'_c, \hat{\boldsymbol{b}}' \leftarrow \mathcal{L}_D$
10. $m',r' \leftarrow \mathcal{L}_G$
11. $\boldsymbol{v}'_c, \boldsymbol{b}' := \texttt{PKE.Enc}(pk,m',r')$
12. return $\boldsymbol{b}' - \hat{\boldsymbol{b}}'$

$\mathcal{H}(m,c)$:

1. if: $\exists K : (m,c,K) \in \mathcal{L}_H$
2. return K
3. $K \leftarrow \mathcal{K}$
4. if: $m = w$
5. QUERY_H := true
6. **abort**
7. $\mathcal{L}_H := \mathcal{L}_H \cup (m,c,K)$
8. return K

$\texttt{Decaps}(pk=(\boldsymbol{b},seed_{\boldsymbol{A}}),c=(\boldsymbol{v}'_c,\boldsymbol{b}'))$:

1. $\mathcal{L}_D := \mathcal{L}_D \cup (\boldsymbol{v}'_c,\boldsymbol{b}')$
2. if: $\exists (m,r) \in \mathcal{L}_G : c = \texttt{PKE.Enc}(pk,m,r)$
3. if: $m = w$
4. return $K := \mathcal{H}'(c)$
5. return $K := \mathcal{H}(m,c)$
6. else:
7. return $K := \mathcal{H}'(c)$

$\mathcal{G}(m)$:

1. if: $\exists r : (m,r) \in \mathcal{L}_G$
2. return r
3. $r \leftarrow \mathcal{U}(\{0,1\}^{l_r})$
4. $\mathcal{L}_G := \mathcal{L}_G \cup (m,r)$
5. return r

Fig. 6: Adversary $\mathcal{B}$ against the SIP0-LWE game

Game G_5: In Game G_5 we change oracle $\mathcal{H}$ to raise the CHAL flag and abort on the query (m^*,c^*), and the oracle $\mathcal{G}$ to raise the CHAL flag and abort on the query m^*. After this game both keys are uniformly random from the point of view of the adversary, hence:

$$\Pr[G_5] = \frac{1}{2}.$$

We construct an adversary $\mathcal{C}$ against the OW-CPA security of PKE as given in Figure 7. During this game, $\mathcal{L}_m$ is a list of queried m values. If the CHAL flag is raised, the adversary has queried on the message m^*, which implies $m^* \in \mathcal{L}_m$, and $\mathcal{C}$ can return $\overline{m} = m^*$, and thus:

$$|\Pr[G_5 = 1] - \Pr[G_4 = 1]| \leq (q_H + q_G)\text{Adv}^{\text{ow-cpa}}_{\mathsf{PKE}}(\mathcal{C}).$$

□

B Security proof of Theorem 3

Proof. The proof essentially proceeds analogous to the proof of Theorem 2. The only difference is in the discussion of Game G_3 and Game G_4. We will therefore only detail these steps.

$\mathcal{C}(pk=(\boldsymbol{b},seed_{\boldsymbol{A}}),c^*)$:

1. $w \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
2. $k \leftarrow \mathcal{K}$
3. $b' := \mathcal{A}^{\mathsf{Decaps}(pk,\cdot),\mathcal{H},\mathcal{G}}(pk,c^*,k)$
4. $m' := \mathcal{L}_H$
5. return m'

$\mathcal{H}(m,c)$:

1. if: $\exists K : (m,c,K) \in \mathcal{L}_H$
2. return K
3. $\mathcal{L}_m := \mathcal{L}_m \cup m$
4. $K \leftarrow \mathcal{K}$
5. if: $m = w$
6. QUERY_H := true
7. **abort**
8. $\mathcal{L}_H := \mathcal{L}_H \cup (m,c,K)$
9. return K

$\mathtt{Decaps}(pk=(\boldsymbol{b},seed_{\boldsymbol{A}}),c=(\boldsymbol{v}'_c,\boldsymbol{b}'))$:

1. if: $\exists (m,r) \in \mathcal{L}_G : c \neq \mathtt{PKE.Enc}(pk,m,r)$
2. if: $m = w$
3. return $K := \mathcal{H}'(c)$
4. return $K := \mathcal{H}(m,c)$
5. else:
6. return $K := \mathcal{H}'(c)$

$\mathcal{G}(m)$:

1. if: $\exists r : (m,r) \in \mathcal{L}_G$
2. return r
3. $\mathcal{L}_m := \mathcal{L}_m \cup m$
4. $r \leftarrow \mathcal{U}(\{0,1\}^{l_r})$
5. $\mathcal{L}_G := \mathcal{L}_G \cup (m,r)$
6. return r

Fig. 7: Adversary $\mathcal{C}$ against the OW-CPA

Game G_3: As in previous proof, we don't check if $m = \overline{m}$ anymore. Remember that during decapsulation the message is calculated as $\overline{m} := \mathtt{decode}(\boldsymbol{v}' - \boldsymbol{v})$, $\boldsymbol{v}' - \boldsymbol{v}$ is exactly the term checked during decapsulation. Therefore we have:

$$\overline{m} := \mathtt{decode}(\overline{\boldsymbol{s}'}\boldsymbol{e} - \overline{\boldsymbol{e}'}\boldsymbol{s} + \overline{\boldsymbol{e}''} + \overline{\boldsymbol{u}} + \mathtt{encode}(m)),$$

which results in the same bound as for the proof of Theorem 2:

$$|P[G_2 = 1] - P[G_1 = 1]| \leq q_G \delta. \tag{6}$$

Game G_4: In Game G_4, the validity check of ETC2 is replaced by the validity check of the FO transformation. As discussed in the proof of Theorem 2, the only difference between Game G_3 and Game G_4 can be observed if an adversary submits a ciphertext that is accepted in Game G_3, but is non-valid in Game G_4.

We use an adversary $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against the SIP-LWE problem, as given in Figure 8, where $\mathcal{H}$, $\mathcal{G}$ and `decaps` are as given in Figure 6. As in the proof of `ETC1`, if the adversary $\mathcal{A}$ has found a difference between both games, one of the ciphertexts in the list $\mathcal{L}_D$ is a differentiating query c_{diff}, which has a corresponding $(\hat{m}', \hat{r}')$ in $\mathcal{L}_G$ so that:

$$(\hat{\boldsymbol{v}}'_c, \hat{\boldsymbol{b}}') \neq \mathsf{PKE.Enc}(pk,\hat{m},\hat{r}),$$

due to the check rejection in G_4 and:

$$\boldsymbol{b}'\boldsymbol{s} - \boldsymbol{v}'_c = \boldsymbol{z} = \hat{\boldsymbol{b}}'\boldsymbol{s} - \hat{\boldsymbol{v}}'_c,$$

$\mathcal{B}(pk=(\boldsymbol{b},seed_{\boldsymbol{A}}))$:

1. $m^* \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
2. $r^* := \mathcal{G}(m^*)$
3. $c^* := \mathsf{PKE.Enc}(pk, m^*; r^*)$
4. $k_0 := \mathcal{H}(m^*, c^*)$
5. $k_1 \leftarrow \mathcal{K}$
6. $b \leftarrow \mathcal{U}(\{0,1\})$
7. $w \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
8. $b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(pk,\cdot),\mathcal{H},\mathcal{G}}(pk, c^*, k_b)$
9. $\hat{\boldsymbol{v}}'_c, \hat{\boldsymbol{b}}' \leftarrow \mathcal{L}_D$
10. $m', r' \leftarrow \mathcal{L}_G$
11. $\boldsymbol{v}'_c, \boldsymbol{b}' := \mathsf{PKE.Enc}(pk, m', r')$
12. return $(\boldsymbol{b}' - \hat{\boldsymbol{b}}', \boldsymbol{v}'_c - \hat{\boldsymbol{v}}'_c)$

Fig. 8: Adversary $\mathcal{B}$ against the SIP-LWE game

due to the acceptance in G_3, where $(\boldsymbol{b}', \boldsymbol{v}'_c) := \mathsf{PKE.Enc}(pk, m'; r')$.

We know that $c \neq \hat{c}$ as the ciphertext $\hat{c}$ would have been accepted in G_4 otherwise, and thus or $\boldsymbol{b}' \neq \hat{\boldsymbol{b}}'$, or $\boldsymbol{v}' \neq \hat{\boldsymbol{v}}'$. In the latter case, the check of Game G_3 still needs to be fulfilled and thus $\boldsymbol{b}' \neq \hat{\boldsymbol{b}}'$. The adversary can thus construct a solution to the SIP-LWE problem by outputting $\boldsymbol{a}_* := \boldsymbol{b}' - \hat{\boldsymbol{b}}' \neq 0$ and $\boldsymbol{b}_* := \boldsymbol{v}' - \hat{\boldsymbol{v}}'$, and we can conclude:

$$|P[G_4 = 1] - P[G_3 = 1]| \leq q_D q_G \mathrm{Adv}^{\text{SIP-LWE}}_{R,l,\chi_s,\chi_e,q_D}(\mathcal{B}).$$

The rest of the proof proceeds identically to the proof of Theorem 2.

□

C ET Transformation in the Quantum Random Oracle Model

In this section we discuss the security of our ET transformation in the Quantum Random Oracle model (QROM), where a quantum IND-CCA adversary against $\mathsf{KEM}_{\mathsf{ET}}$ can call the hash functions on a superposition of queries. We refer to [NC11] for basic on quantum computing. We give an high level description of the security proof using the semi-classical oracle technique introduced by Ambainis et al. [AHU19].

First we recall the following fundamental results.

Lemma 1 (Simulating the random oracle, [Zha12]). *Any quantum algorithm $\mathcal{A}$ making q quantum queries to a random oracle $\mathcal{O}$ can be efficiently simulated by a quantum algorithm $\mathcal{B}$ which has the same output distribution but makes no queries.*

In particular it was shown in [Zha12], that is enough to simulate and answer random queries with evaluations of $2q$-wise independent functions, given that q is the number of queries to the oracle $\mathcal{O}$. In particular, we have the following lemma.

Lemma 2 ((Preimage search in a random function, [ARU14], Lemma 37). *Let $\gamma \in [0,1]$. Let Z be a finite set. Let $N_1 : Z \to \{0,1\}$ be the function defined as follows. For each z,*

$$\begin{cases} N_1(z) = 1 \text{ with probability } p_z \, (p_z \leq \gamma), \\ N_1(z) = 0 \text{ otherwise .} \end{cases}$$

Let N_2 such that $\forall z, N_2(z) = 0$. If an oracle algorithm $\mathcal{A}$ makes at most q quantum queries then

$$|\Pr[b = 1 : b \leftarrow \mathcal{A}^{N_1}] - \Pr[b = 1 : b \leftarrow \mathcal{A}^{N_2}]| \leq 2q\sqrt{\gamma}.$$

In particular, the probability of $\mathcal{A}$ finding a z such that $N_1(z) = 1$ is at most $2q\sqrt{\gamma}$.

In the classical ROM setting, reductions simulate the random oracle so to learn the adversary's queries. However, in the quantum setting, this is no longer trivial as measuring or recording queries would change the adversary's state.

A very useful tool, used to solve the problem of reprogramming random oracles, is the *One-way to hiding technique* (OW2H). There are essentially three different versions of the OW2H lemma, the original formulation introduced in [Unr14], the semi-classical OW2H lemma [AHU19] that we use in this section, and the double-sided version [BHH+19,KSS+20] which allows to achieve tighter bounds. In particular, Kuchta et al. [KSS+20] presented a new version of the double sided OW2H lemma, which does not suffer from the square root loss in the reduction that we can find in previous works. A complete and detailed security proof of our ET transformation in the QROM using this new methodology is orthogonal to the main result of this current paper and is therefore left to future work.

Proof strategy. Given a δ-correct PKE, to bound the advantage of any adversary $\mathcal{A}$ against the IND-CCA security of $\mathtt{KEM}_{\mathtt{ET}}$ we proceed according to a standard game hoping technique, following the same proof strategy described in [JZM19].

Let $\mathcal{A}$ be an adversary against the IND-CCA security of KEM, issuing at most q_D classical queries to the decapsulation oracle DECAPS , at most q_G queries to G and at most q_H queries to H. Given (pk, sk) and $m \in \mathcal{M}$, let

$$\mathcal{R}_{\text{good}} := \{r \in \mathcal{R} : \mathsf{PKE.dec}(sk, \mathsf{PKE.Enc}(pk, m; r)) = m\}.$$

Let Ω_G be the set of all functions $G : \mathcal{M} \to \mathcal{R}$. Let G' be a random function such that $G'(m)$ is sampled according to the uniform distribution in $\mathcal{R}_{\text{good}}$. Moreover, let $\delta(pk, sk, m) = \frac{|\mathcal{R}_{\text{bad}}(pk, sk, m)|}{|\mathcal{R}|}$, $\delta(pk, sk) = \max_{m \in \mathcal{M}} \delta(pk, sk, m)$ and $\delta = \mathbf{E}[\delta(sk, pk)]$, where the expectation is taken over $(pk, sk) \leftarrow \mathsf{PKE.Gen}$. Let Ω_H and $\Omega_{H'}$ be the set of all functions $H : \mathcal{M} \times \mathcal{C} \to \mathcal{K}$ and $H' : \mathcal{C} \to \mathcal{K}$, respectively.

The first three games proceed exactly as in the proof of [JZM19]. In particular, GAME G_0 is exactly the IND-CCA security game of $\mathsf{KEM}_{\mathsf{ET}}$ (Figure 9); in GAME G_1, we change the DECAPS oracle so that $H'(c)$ is returned instead of $H(w, c)$, when c is an invalid encapsulation. In GAME G_2, we replace G by $G' \leftarrow \Omega_{G'}$, that samples from $\mathcal{R}_{\text{good}}$ uniformly at random.

GAME G_0

1. $(pk,sk) \leftarrow \mathsf{KeyGen}()$
2. $G \leftarrow \Omega_G$
3. $H' \leftarrow \Omega_{H'}$, $\bar{H} \leftarrow \Omega_H$
4. $w \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
5. $m^* \leftarrow \mathcal{U}(\{0,1\}^{l_m})$
6. $r^* := G(m^*)$
7. $c^* := \mathsf{PKE.Enc}(pk,m^*;r^*)$
8. $k_0 := H(m^*,c^*)$
9. $k_1 \leftarrow \mathcal{K}$
10. $b \leftarrow \mathcal{U}(\{0,1\})$
11. $b' \leftarrow \mathcal{A}^{H,G,\mathsf{Decaps}}(pk,c^*,k_b)$
12. **return** $b = b'$

DECAPS
$(sk=(\boldsymbol{s},\boldsymbol{e}),pk=(\boldsymbol{b},\mathrm{seed}_{\boldsymbol{A}}),c=(\boldsymbol{v}'_c,\boldsymbol{b}'))$:

1. $t = (\boldsymbol{b}'\boldsymbol{s},\boldsymbol{v}'_c)$
2. $\overline{m} := \mathsf{PKE.Dec}(sk,c)$
3. $\overline{r} := G(\overline{m},pk)$
4. **if**: $t = \mathsf{KEM}_{\overline{\mathrm{ET}}}.\mathsf{GenET}(sk,pk,\overline{m};\overline{r})$
5. **return** $K := H(\overline{m},c)$
6. **else**:
7. **return** $K := H(w,c)$

$H(m,c)$

1. **return** $\overline{H}(m,c)$

Fig. 9: Game G_0 - IND-CCA security of $\mathsf{KEM}_{\mathsf{ET}}$

In GAME G_3, we replace the check in the DECAPS oracle, so that the error term is replaced by the FO validity check, and then use the SIP0-LWE problem to bound the advantage of $\mathcal{A}$ to distinguish between G_2 and G_3. In the next game, GAME G_4, we change H so that if the adversary queries it on input (m,c) such that $g(m) = \mathsf{PKE.Enc}(\mathsf{pk},m;G(m)) = c$, the response will be $H_1(c) = H_1 \circ g(m)$.

The rest of the proof will essentially proceed like in [JZM19], by applying the OW2H lemma to reprogram the random oracle.